

METHOD AND ARCHITECTURE FOR BUILDING CLIENT-SERVER APPLICATIONS

FIELD OF THE INVENTION

[0001] The present invention relates to client-server applications.

BACKGROUND OF THE INVENTION

[0002] The Internet has grown rapidly in recent years and has now become an accepted and popular way to communicate and to conduct many types of business activities. For example, the Internet is commonly used for e-mail, searching for information (e.g., news, weather, sports, etc.), and for buying and selling goods online.

[0003] The web browser has become the software application of choice as the gateway to the Internet. A typical browser (on the client side) interacts with a server to request and retrieve information over a network (e.g., the Internet). The browser operates in a "sandbox" environment that prevents programs downloaded by the browser from having access to the clients' file system.

[0004] In operation, the browser requests a hyper-text markup language ("HTML") page from a server. The server fulfills this request by sending this page back to the browser. Once this is done the connection is dropped and the server continues serving requested pages in response to client requests. As a result, the hyper-text transmission protocol ("HTTP") is considered "stateless" since it does not keep track of the pages being viewed by the client. Because of the "stateless" nature of the HTTP protocol, the browser in most cases only supports the presentation of information, with the application logic wholly residing in the server.

[0005] The worldwide web has evolved from its basic concept of serving static HTML pages to providing rich, dynamic web pages and information derived from web-based databases. Many techniques have been developed to offer the dynamic content users see on their browsers using scripting languages (e.g., JavaScript) and server side programs built using PHP, C++, Java™, and active server pages.

[0006] Although there are a number of techniques available to provide the application functionality on the browser-end, building web-based client server applications has been challenging. For example, applications including “applets” are capable of being downloaded by the browser, but bandwidth restrictions limit the size of the applet that can be downloaded. Technology also exists that allows documents using Microsoft Word™, for example, to be embedded into the browser. However, this technology is proprietary. A tunneling protocol that rides on top of the HTTP where more sophisticated client-server instruction sets can be used has also been proposed. However, to date, no software is available on the market that implements this technological approach.

[0007] Creating effective web-based client-server applications has been difficult. Standard applications such as a word processor typically reside on the users’ desktop. Once open, these applications load into the systems’ program memory and are run “in process” (i.e., the application will respond instantaneously to any user changes while the user is adding or modifying the document). In the case of client-server systems, part or all of the application generally resides on the remote server where an “in-process” application can communicate with an “out-of-process” application on a remote server. However, for these applications to talk to each other effectively in a distributed environment requires a detailed understanding of network protocols as well as operating systems. As a result, several specifications have emerged to enable distributed technology, namely the distributed component object

model ("DCOM") used for Microsoft Windows™ and common object request broker architecture ("CORBA™") for unix. Javascript, with its remote method invocation ("RMI") capabilities, also provides scalable distributed software in the form of enterprise java beans ("EJB").

[0008] Client-server applications are generally divided into three components. The presentation logic of the application most often resides on the client, the application logic is typically divided between the client and the server, while the data management layer usually resides on the server. This complicates the design of client-server applications because the entire application needs to be maintained in a known state over the distributed environment namely through several computer systems.

[0009] What is needed is a new method and architecture for building reliable and efficient client server applications around a browser that overcomes the limitations inherent in the prior art.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The present invention is illustrated by way of example, and not limitation, in the figures of the accompanying drawings, wherein:

[0011] Figure 1 shows a block diagram of a conventional client-side browser interacting with a server to request and retrieve information according to the prior art.

[0012] Figure 2A shows a conventional three-tiered client-server application according to the prior art.

[0013] Figure 2B shows multiple objects held in the three-tiered client-server application of Figure 2A.

[0014] Figure 3 shows a page for creating a database form on a client-side browser according to one embodiment of the present invention.

[0015] Figure 4 shows an example form created by a user on a client-side browser according to one embodiment of the present invention.

[0016] Figure 5A shows application state variables held in a database table as a component on a server according to one embodiment of the present invention.

[0017] Figure 5B shows how state maintenance is achieved in a practical system using the model of Figure 5A.

[0018] Fig. 5C shows a typical client server interaction showing the process of state information held in the tables.

[0019] Fig. 5D shows an extension of the previous model where application code can also reside in tables.

[0020] Fig. 5E shows a scalable distributed system employing the methods of state, program code and other types of information held in a common storage space.

[0021] Figure 6 shows a program listing in JavaScript used to create a text box in the design phase of a database form using cookies according to one embodiment of the present invention.

[0022] Figure 7 shows a program routine executed during redrawing of a database form following a "state" change held in a cookie string according to one embodiment of the present invention.

[0023] Figure 8 shows concatenating a cookie variable to hold the form schema and customer information for storage and retrieval from a database on a server according to one embodiment of the present invention.

[0024] Figure 9 shows creating a JavaScript array on a server for use on a client computer according to one embodiment of the present invention.

[0025] Figure 10 is a flow chart showing the creation of a new database form on a client computer using a web browser and submitting the form to a server according to one embodiment of the present invention.

[0026] Figure 11 is a flow chart that shows reconstructing a database form on a server and submitting the form to a client computer according to one embodiment of the present invention.

[0027] Figure 12 is an example of a computer system on which the present techniques can be implemented.

[0028] Figure 13 is a block diagram of a server computer system connected to multiple client computer systems.

005793.P001

DETAILED DESCRIPTION

[0029] The present invention provides a system and method for building a scalable client-server. A unique database application has been built using the methods and processes described. Although well suited for use in database applications, the system and method described may also be extended to include other networked applications using a browser, such as word processor applications, project management, accounting, and resource planning. The system and method is also applicable to any generic based client-server system where the state of a distributed system is known at any point in time. Accordingly, the specification and drawings are to be regarded in an illustrative, rather than a restrictive sense.

[0030] Referring now to Figure 1 there is shown a block diagram of a conventional client-side browser interacting with a server to request and retrieve information according to the prior art. Using a browser 104, a client computer 101 requests a hyper-text transfer protocol ("HTTP") page 102 from a web server 103 through a network (e.g., the Internet) 109. The client computer 101 includes an operating system 105 and a file system 106. The browser 104 is able to communicate with these systems and also with the web server 103 in a manner known in the art. Once the client computer 101 requests the HTTP page 102, the web server 103 accesses active server pages 107 and/or the database 108 and returns the HTTP page 102 to the browser 104 through the network connection 109. Once this is accomplished, the connection is dropped and the server 103 continues serving HTTP pages in response to client requests.

[0031] Figure 2A shows a conventional three-tiered client-server application according to the prior art. The three-tiered client-server application of Figure 2A divides the client-server into three components. The presentation logic 201 almost

always resides on the client 202, the application logic 203 typically resides between the client 202 and the server 204. The format of the application logic 203 held on the client 202 is usually held as an object 206 (software code) made using programming languages such as C++ or Java. In some cases part or all of the application logic 203 can be held in the form of Structured Query Language ("SQL") procedures well known in the art. These applications typically communicate through proprietary protocol developed by makers of the operating system which gives greater functionality to the client-server than applications using the http protocol such as a browser (thin client). In all cases, the application is considered to be a piece of code operating in its entirety and acts as the middle tier link between the client and the data store. The most widely used architecture have clients perform all of the presentation and some of the application logic where objects are developed in stand-alone fashion and serve as the communication link and intelligence between the client 202 and the data 207.

[0032] Figure 2B shows multiple objects held in the three-tiered client-server application of Figure 2A. In the typical client-server application there are a number of objects 220, 230, 240, 250, etc., each operating in its own space and being instantiated or destroyed whenever the need arises. These objects 220, 230, 240, 250, etc., perform specific functions depending on the task requested by the user. Because each object performs in its own space, it is difficult to know the state at a specific time unless all objects 220, 230, 240, 250, etc., are interrogated simultaneously and requested to give information about their state.

[0033] Referring now to Figure 3 there is shown a page for creating a database form on a client-side browser according to one embodiment of the present invention. In operation, a client user accesses a hyper-text markup language ("HTML") pages from a HTTP server (not shown in this view). This is generally

achieved by entering the uniform resource locator ("URL") string from the HTTP server in the web browser on the client computer. It should be appreciated that although the network described herein is the intranet, a Wide Area Network ("WAN"), a Local Area Network ("LAN"), or any other system of interconnections enabling two or more computers to exchange information may be used as well. Further, network may include a wireless network, such that one or more computers may operate over a wireless LAN (WLAN).

[0034] In the embodiment illustrated by Figure 3, the page for creating a database form is requested by the client from the web server by accessing the URL (not shown in this view). The form to create the database is contained in this HTML page. To create the application, "cookies" are used to hold application state variables (not shown in this view). Cookies are a mechanism to allow browsers to store information from a specific domain and are often used to hold specific information such as login data, when the site was last visited, etc. The act of using cookies to store specific information coupled to the act of refreshing the client page allows state information to be held on the client side. This allows for the development of dynamic applications without repeated requests from the client to the server. Each time an application variable is stored (i.e., a piece of information of the client form is changed) it is reloaded on the client computer to reflect the change. At the same time the cookie string is also changed to track this change.

[0035] For example, if a user adds a Text Box 301 to the form, the form will reload and append this variable to the cookie string. In this manner, cookies are used as storage space much in the same way that random access memory ("RAM") is used to hold application state in a central processing unit ("CPU"). In one embodiment, JavaScript is used to capture the new form variable and append the new variable to the existing cookie. Of course, perl, Vbscript, PHP, or any other

scripting programs that are executable on a browser may also be used. Because all the application logic resides in the HTML page, the client does not need to make repeated requests to the server during the database form creation thus reducing network traffic. The method of cookies storage to hold application state can be extended to include other forms of software containers. This could be in the form of client side arrays where a dynamic storage space can be allocated for holding state information and accessible through client side scripts.

[0036] In the example shown in Figure 3, there are four options the user can use to create a customized database form (i.e., the Text Box 301, the Check Box 302, the Option Button 303, and the Select Box 304). Of course, other template selections (not shown in this view) may be used as well. A user may select the appropriate form type, and enter a Field Name 305, 306, 307, and 308 and a Sub Name 309, 310, and 311 for the form type. By clicking the Add buttons 312, 313, 314, and 315 for the form types, the form type is appended to the end of the form that is being created (i.e., the form is created using cookies storing information relevant to that form on the Text Box 301). Any combination of Text Boxes 301, Check Boxes 302, Option Buttons 303, and Drop-Down list boxes (not shown in this view) can be added in this manner. An input box 316 allows the user to control the size of the record in the database using the scroll down icon 317. When the form is completed, the user clicks the Make Active button 318 to submit the form to the server (not shown in this view) where it is stored for further use. The user can simply access this form to store and retrieve records from the server using a web browser.

[0037] Referring now to Figure 4 there is shown an example form created by a user on a client-side browser according to one embodiment of the present invention. The form in Figure 4 is an example video form created using the forms

screen of Figure 3. Of course, a wide variety of forms for storing different types of information may be created using the forms screen. To view a record (such as the record illustrated by Figure 4), a client request is transmitted to the server and the server sends the appropriate record back to the client. In one embodiment of the invention a user may specify a range of data he wants to look at. The server receives the data range request and packages the data in a JavaScript array (of course, other types of software arrays can be used as well) and then sends this data back to the client where the client can handle it directly without further requests to the server. In addition, JavaScript can be used to perform calculations or analysis of the data through the JavaScript data array. This is an efficient way of retrieving data and placing application logic on the client side that minimizes network usage.

[0038] Figure 5A shows application state variables held in a database table as a component on a server according to one embodiment of the present invention. The architecture illustrated in the embodiment of Figure 5A makes use of a database table 510 to perform state maintenance. After creating and submitting the form described earlier the client can request to operate on this new form (i.e., add or retrieve information through this form) much like any other database application.

[0039] State variables of the application session are maintained through a table held in the database 510 shown in Fig. 5A. Objects 501, 502, 503, 504, etc., are held in table format 510 and then reconstructed on HTML through cookie variables held in a string (not shown in this view). In addition, a master (system) controller (e.g., a web server) 505 oversees and monitors the objects' state by interrogating the table information 510 where the objects 501, 502, 503, 504, etc., hold their state. The objects 501, 502, 503, 504, etc., transmit state information to the table 510, so a snapshot of the distributed environment is always known. When a client requests a form, the server calls up only that string. The string is then

reconstructed on the client end (e.g., the JavaScript reconstructs the form using cookies).

[0040] Fig 5B shows an implementation of the model discussed above where the web server handles the objects and state information held in tables or other software containers. A client 515 communicates with the web server via a network (not shown in this view). In the context of the database application, Fig. 5C shows the process of state maintenance using this new model. A client requests a form (processing block 520) from a server (processing block 525). The client's session begins by creating an entry in the table that identifies the client as a user and identifies the form that the client needs to utilize (processing block 530). The client requests stored data using this form (processing block 535). The system reserves sufficient space within the table to allow the retention of state variables within this current session (processing block 540). Once the session has expired (i.e., when the user has finished with the application), the state variables may be archived for future reference (processing block 545) or discarded. The client works with the returned data collection packaged as a JavaScript array (processing block 550). Since databases are only limited by the capacity of the physical drive, the archiving of state information can be very powerful for troubleshooting since the historical state of objects is never lost. As the session progresses, state information is continuously added, updated or deleted within the reserved space for this session.

[0041] A second architecture can be realized from the previous model. Fig. 5D shows application codes 555 and 556 residing in the table or other software container in addition to state variables 557 and 558. The code may be executed line by line (interpreted method) whereas the rows in the table can represent each line of code. An object handler may be responsible for the execution of the code.

Alternatively, the resident code may be a binary file in which case the table may

retain the whole code in the table with a pointer to it. Code may be extracted and executed through the object handler from requests by the web server.

[0042] Fig. 5E shows a complete distributed architecture based on these methods. The architecture consists of a load balancer 560, a single middle tier multi-processor (MP) system 561 or 571 and a backend redundant storage system 562 for holding the database. The load balancer 560 redirects client traffic according to the load distribution of the MP systems 561 and 571 and selects the MP system with the minimum load. A web server 562 or 563 responds to the clients' request and pulls the appropriate pages from the storage system. The redundant storage system 562 holds the database and its content to reliably maintain and operate the whole distributed system. This will typically include all object code 566 and 572, web pages and other static and dynamic information related to the system as a whole. A request for an object is made via a web page through the web server 562 or 563. A database instance 564 or 565 pulls the appropriate code from the table in the storage system and is instantiated in the specific MP system 561 or 571. The request made by the database instance 564 or 565 may be in the form of SQL queries and handed back to the web server 562 or 563 for processing. State variables for this object are maintained in the common table and are accessible by any of the MP systems 561 or 571.

[0043] Communication between objects between two MP systems 561 or 571 is carried out through the state variables maintained within the common tables. By retaining all relevant information within a database or distributed software container a new level of software abstraction is achieved. The dependence on a specific file system of the operating system is avoided since information is now stored through the common database software component. Fig. 5E illustrates a parallel database that may implement the architecture of the present invention. In addition to using

databases, other distributed type of software containers may also be used. The middle tier shown in Fig. 5E are CPU systems where all requests, object instantiation and queries are executed.

[0044] A relational database may be used for mapping the file system of an individual machine including other components such as drivers, registry information etc. The storage of application state within the tables is useful in many applications, such as chat. In one example, the chat text between two instantiated objects (two people engaged in a chat) are held in the table and returned to the clients from the table (not shown in this view).

[0045] Figure 6 shows a program listing in JavaScript used to create a text box in a design stage of a database form according to one embodiment of the present invention. Lines 5 and 6 show the values chosen by the user for a Text Box name and Text Box size, respectively. Line 12 increments by one the total number of application variables stored in the cookies. The new cookies variables are created (line 14) and are written to the cookie container of the clients' machine (line 15). Immediately following this step, the browser window is redrawn to include the new Text Box requested by the user (line 17). Repeating this step allows additional form elements to be added. The browser does not make any requests to the server during the design phase and it is only after the form is completed that the information is sent to the server. It should be noted that the steps for deleting a form row are similar to the steps for creating a row except that the total number of cookies (i.e., application variables) are reduced by one.

[0046] Figure 7 shows a program routine executed during redrawing of a database form based on new information held in a cookie string according to one embodiment of the present invention. The cookie variables are placed in a two dimensional array (data_sets) for easier handling. Line 1 loops through the cookie

variables and builds an HTML string containing the rows of the form. When this is completed, the whole string is written to the client (line 29) to display the new page.

[0047] Figure 8 shows concatenating a cookie variable to create a database form schema according to one embodiment of the present invention. When submitting the new form to the server for storage, the cookie variables are combined into a long string (lines 1-4). The string contains all the information necessary to rebuild this form. The separator '^AAA^' is used to distinguish each of the form elements. Lines 6-13 show the code that inserts the cookie variables along with the users' details into the server database.

[0048] Figure 9 shows creating a JavaScript array on a server for use on a client computer according to one embodiment of the present invention. As described herein, to add a record a user calls up the form and submits the record to a server where it is stored in a database. To view a collection of records, the user sends the specified data range request to the server. The server-side script retrieves the data from the database and creates the JavaScript array prior to sending it to the client. Data held in the array allows the user to scroll through the collection on the client-side without making repeated requests to the server. For example, the user can click the back and forward keys to scan through the records held in the JavaScript array.

[0049] Figure 10 is a flow chart that shows the steps of creating a new database form on a client computer using a web browser and submitting the form to a server according to one embodiment of the present invention. A client user accesses an HTML page from a server (processing block 601). client user creates a customized database form using the browser by selecting various templates and entering various field names for each record in the database (processing block 602). The user submits the form to the server using the network connection (processing

block 603). The server stores the form as a string in a table on the server (processing block 604).

[0050] Figure 11 is a flow chart that shows the steps of reconstructing a database form on a server and submitting the form to a client computer according to one embodiment of the present invention. A server receives a client request to transmit a particular form stored on the server (processing block 606). The server calls up the cookie string holding the application state variables for the particular form and transmits the cookie string to the client computer over the network (processing block 607). The client computer reconstructs the form using cookies (processing block 608). The server maintains the state of the running application through state variables held in a database table (processing block 609).

[0051] Figure 12 is an example of a computer system on which the present techniques may be implemented according to one embodiment of the present invention. The computer system 700 includes a processor 702 coupled through a bus 701 to a random access memory (RAM) 703, a read only memory (ROM) 704, and a mass storage device 705. Mass storage device 705 could be a disk or tape drive for storing data and instructions. A display device 706 for providing visual output is also coupled to processor 702 through bus 701. Keyboard 707 is coupled to bus 701 for communicating information and command selections to processor 702. Another type of user input device is cursor control unit 708, which may be a device such as a mouse or trackball, for communicating direction commands that control cursor movement on display 709. Further coupled to processor 702 through bus 701 is an input/output (I/O) interface 710 which can be used to control and transfer data to electronic devices connected to computer 700, such as other computers, tape records, and the like.

[0052] Network interface device 711 is coupled to bus 701 and provides a physical and logical connection between computer system 700 and the network medium (not shown in this view). Depending on the network environment in which computer 700 is used, this connection is typically to a server computer, but it can also be to a network router to another client computer. Note that the architecture of Figure 12 is provided only for purposes of illustration, and that a client computer used in conjunction with the present invention is not limited to this specific architecture.

[0053] Figure 13 is a block diagram of a server computer system coupled to a client computer system. Client computer 801 is coupled to a server computer 802 through network 803. The network interface between client 801 and server 802 may also include one or more routers, such as routers 803 and 804, which serve to buffer and route the data transmitted between client 801 and server 802. Network 803 may be the Internet, a Wide Area Network ("WAN"), a Local Area Network ("LAN"), or any combination thereof. Network server 802 contains application programs and/or data that are accessible over the network by other network stations, such as network client 801. In one embodiment of the present invention, network server 802 is a World-Wide Web ("WWW") server, which stores data in the form of 'web pages' and transmits these pages as HTML files over the Internet network 803 to a network client 801. To access these files, network client 801 runs a web browser (not shown in this view), which is simply an application program for accessing and providing links to web pages available on various Internet sites. In a typical Internet client-server environment, the client computer accesses the Internet through a single point of contact, commonly referred to as an Internet Service Provider (ISP) or on-line service provider.

[0054] In the foregoing, a system and method has been described for client-based form creation using a browser. Although the present invention has been described with reference to specific exemplary embodiments, it should be understood that numerous changes in the disclosed embodiments can be made in accordance with the disclosure herein without departing from the spirit and scope of the invention. The preceding description, therefore, is not meant to limit the scope of the invention. Rather, the scope of the invention is to be determined only by the appended claims and their equivalents.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
222